N80-21373

Arithmetic Techniques Used in the High-Speed Front End of The Multimegabit Telemetry Modem

E. R. Wechsler
Communications Systems Research Section

~~Two~~ methods of processing two's complement binary data are presented. They were used in the design of the High-Speed Front End of the Multimegabit Subsystem. The first method is for rounding with zero error in the mean and is useful when large numbers of rounded values are to be accumulated.

The second method consists of attaching a "1" as a least significant bit to the output word of an A-D converter, thus processing it as an odd number with one more bit. This compensates for the negative bias that A-D converters have when their zero is set for symmetrical positive and negative output ranges as is customary.

When used with emitter coupled logic circuits, these methods result in simpler logic design.

I. Introduction

A Multimegabit Subsystem is currently being developed to increase the telemetry data rate of the DSN to 30 megasymbols per second. The high-speed front end (HSFE) is a portion of the Multimegabit Subsystem which converts quadrature analog signals into digital signals for further processing by the detection, estimation, and synchronization assemblies of the subsystem. The incoming analog signals are converted at rates of up to 64 MHz using monolithic 4-bit quantizers newly developed by Advanced Micro Devices, Inc. The following discussion applies to A-D converter output formatting and rounding techniques used in the HSFE which are applicable to any digital system.

II. Selection of the Rounding Technique for Use in the HSFE

Since the data supplied by the HSFE is to be further processed in the Multimegabit System where it is accumulated, the technique used for rounding must avoid generating unwanted offsets. The assumption is made that all numbers which become equal after rounding are equally probable. A useful observation is that in the two's complement representation, which is used throughout the system, all the bit positions can be considered to have positive weights except for the sign bit which is a negative weight as shown below for an N-bit integer:

Bit label $B_{N-1}, B_{N-2}, \ldots, B_k, B_{k-1}, \ldots, B_1, B_0$

Bit weight $-2^{N-1}, 2^{N-2}, \ldots, 2^k, 2^{k-1}, \ldots, 2^1, 2^0$

Whenever it is necessary to present a binary number in a shorter format, there are several methods which can be selected, such as truncation, conventional rounding, and rounding with the OR function. In what follows these methods will be compared, and it will be explained why rounding with the OR function was chosen for the HSFE.

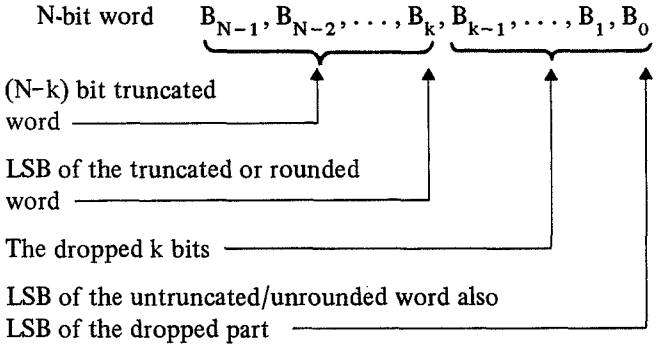
A. The Truncation Method

The truncation method consists of dropping the k least significant bits from the binary word without any further correction. As mentioned earlier, in the two's complement representation the bits are considered positive, so the truncated value will always be equal to or less than the original value which will produce a negative mean error as shown in Fig. 1. If k bits are truncated, then the mean error $\tilde{\epsilon}$ will be:

$$\tilde{\epsilon} = -\frac{0+1+\cdots+(2^k-1)}{2^k} = \left[-2^{-1}+2^{-(k+1)}\right]2^k$$

$$= \left[-1/2+2^{-(k+1)}\right]A$$

where $A = 2^k$ is the weight of the least significant bit (LSB) of the truncated number as shown below:


N-bit word
$B_{N-1}, B_{N-2}, \ldots, B_k, B_{k-1}, \ldots, B_1, B_0$
(N-k) bit truncated word
LSB of the truncated or rounded word
The dropped k bits
LSB of the untruncated/unrounded word also LSB of the dropped part

Assuming infinite resolution for the original number, we get the mean error:

$$\tilde{\epsilon} = -\frac{A}{2}$$

The maximum error:

$$\epsilon_{max} = -A$$

The variance:

$$\sigma^2 = \frac{A^2}{12}$$

B. The Conventional Rounding Method

This method consists of truncating k least significant bits and then arithmetically adding one LSB (in the new truncated format) to the truncated number when the dropped part is equal to or larger than one-half of one LSB in the truncated format and leaving it unchanged otherwise.

Rounding off k bits will produce a mean error of:

$$\tilde{\epsilon} = \frac{-\left[1+\cdots+(2^{k-1}-1)\right]+\left[1+\cdots+2^{k-1}\right]}{2^k}$$

$$= 2^{-1}$$

In this case, the mean error is positive but it is just one-half of one LSB of the dropped part. Since one LSB of the rounded word is $A = 2^k$, the mean error is:

$$\tilde{\epsilon} = \frac{A}{2^{k+1}}$$

Rounding requires hardware for adding the LSB. In addition, when the mean error is to be made zero, even more hardware is needed.

Assuming infinite resolution for the original number we get for the maximum error:

$$\epsilon_{max} = \pm\frac{A}{2}$$

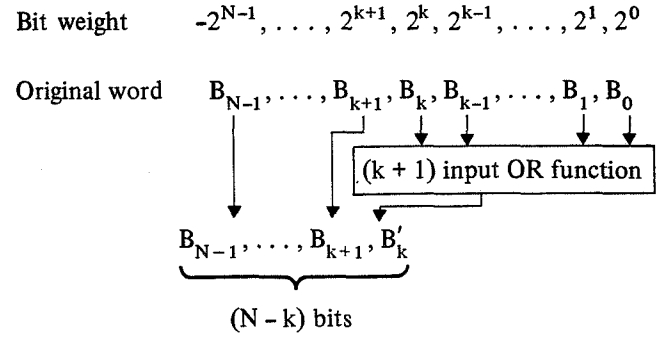
as shown in Fig. 2.

The variance is:

$$\sigma^2 = \frac{A^2}{12}$$

C. Rounding With the OR Function

This method consists of truncating k + 1 bits and then attaching a bit in the least significant position, which is an OR function of the bits that were dropped, thus eliminating just k-bit positions as shown below:


Bit weight
$-2^{N-1}, \ldots, 2^{k+1}, 2^k, 2^{k-1}, \ldots, 2^1, 2^0$
Original word
$B_{N-1}, \ldots, B_{k+1}, B_k, B_{k-1}, \ldots, B_1, B_0$
(k + 1) input OR function
$B_{N-1}, \ldots, B_{k+1}, B'_k$
(N - k) bits

The mean error due to the truncation of k + 1 bits is:

$$\tilde{\epsilon}_1 = -(2^{k+1} - 1)\, 2^{-1}$$

We will attach a "1" as a least significant bit in M cases where M is to be determined as follows:

The weight of this bit is 2^k so the average value of this correction bit will be:

$$\tilde{\epsilon}_2 = \frac{M \cdot 2^k}{2^{k+1}} = \frac{M}{2}$$

In order to have a zero mean error we must have:

$$\tilde{\epsilon}_1 + \tilde{\epsilon}_2 = 0$$

or:

$$\frac{2^{k+1} - 1}{2} = \frac{M}{2}$$

Therefore:

$$M = 2^{k+1} - 1$$

which means that in all cases but one, the attached LSB will be a "1."

The most convenient logic function of the dropped bits turns out to be the OR because it is "0" when the dropped part is zero and the variance is kept minimum. The mean error is zero by design. Assuming infinite resolution, we get the maximum error:

$$\epsilon_{max} = \pm A$$

as shown in Fig. 3.

The variance is:

$$\sigma^2 = \frac{A^2}{3}$$

In conclusion, this method produces maximum errors as large as those due to truncation and an RMS noise twice as large as that due to truncation or rounding. Its advantages are that no adders are required to round and just one OR function is required thus reducing considerably the time delays which in emitter coupled logic circuits can be implemented by simply wire ORing gate outputs.

III. Formatting the Output of the A-D Converter

Aspects of the previous discussion can also be applied to formatting the output of an A-D converter in order to eliminate a residual mean error. We can consider an A-D to be of infinite resolution but supplying a truncated output which has the disadvantage of a negative mean error as shown in Fig. 4 for a three-bit converter.

Applying the OR rounding to an infinite resolution binary word we get a "1" in the LSB location all the time. This is a method of cancelling the -1/2 LSB offset produced by the negative mean error of the A-D and it consists of attaching the "1" all the time on the least significant position, thus increasing the number of bits by one.

It is not necessary to actually supply this LSB as long as its effect is accounted for in subsequent operations. For example, if two numbers have to be added, all we have to do is provide a "1" to the carry input of the adder.

Another feature is generating the two's complement without using an adder. This is done by keeping the LSB a "1" and making the one's complement of the rest of the word, a great advantage in high-speed circuits required in the HSFE.

In addition, the positive and negative ranges of representation are symmetrical as shown in Fig. 5 for a three-bit case, with a "1" attached. This eliminates the problem of unequal ranges when compensating for average zero offset by trimming the A-D references as shown in Fig. 6. This asymmetry becomes very pronounced for A-D converters with few bits as the case of only 4 bits in the HSFE.

An interesting property which results when using odd numbers is that their squares are multiples of eight plus one. For example, let the odd number be: $2m + 1$. Its square will be:

$$(2m + 1)^2 = 4m^2 + 4m + 1 = 4m(m + 1) + 1$$

The product $m(m + 1)$ has to be even, so the first term is a multiple of eight. The square of an odd number in binary representation always ends in 001 which therefore need not be supplied physically.

IV. Conclusion

Two methods of processing binary data in the two's complement were presented, which are used in the High-Speed Front End of the Multimegabit project. The first one is an easy way of rounding, which holds the mean error at zero. The second one allows the easier processing of data from A-D converters while keeping a zero average error (bias) and symmetrical positive and negative ranges.

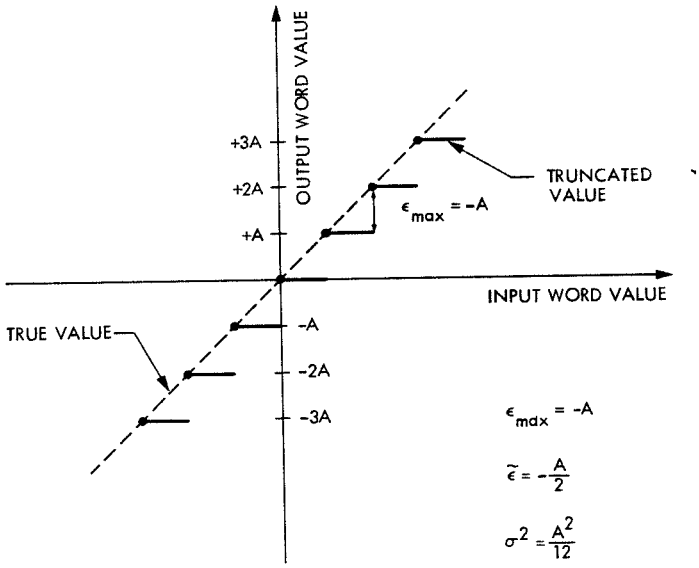


Fig. 1. The truncation method

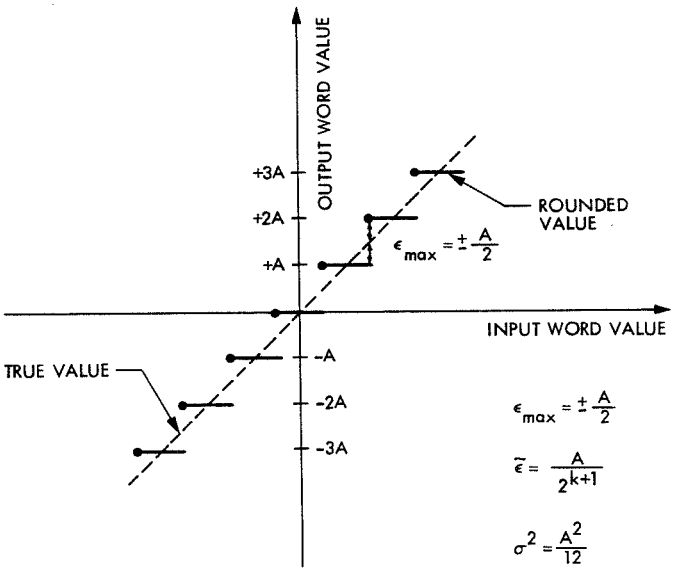


Fig. 2. The rounding method

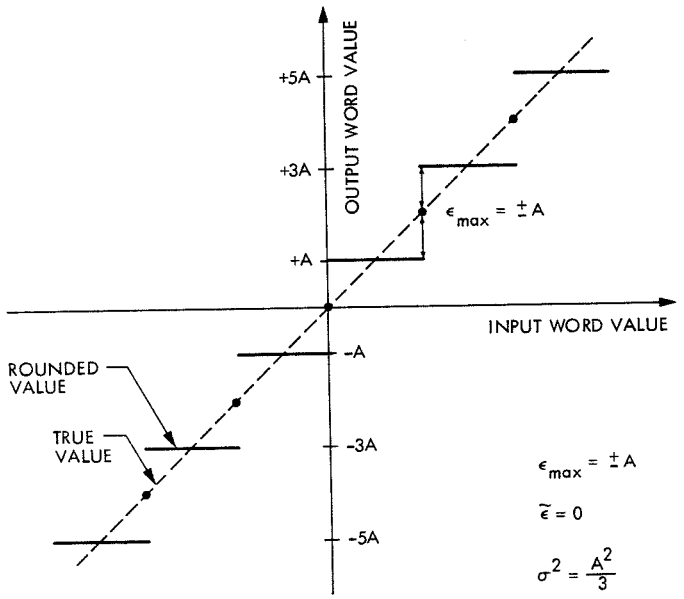


Fig. 3. Rounding with the OR function method

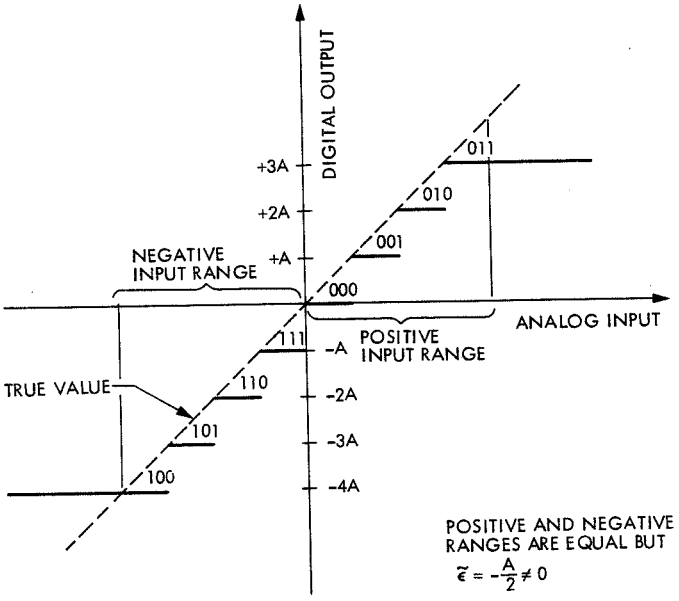


Fig. 4. The A-D converter average error

ORIGINAL PAGE IS
POOR QUALITY

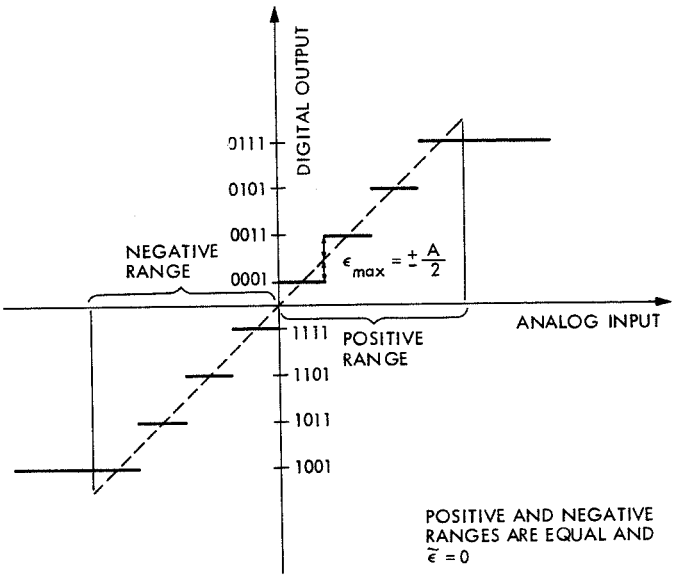


Fig. 5. Output of A-D converter when the odd number method is used

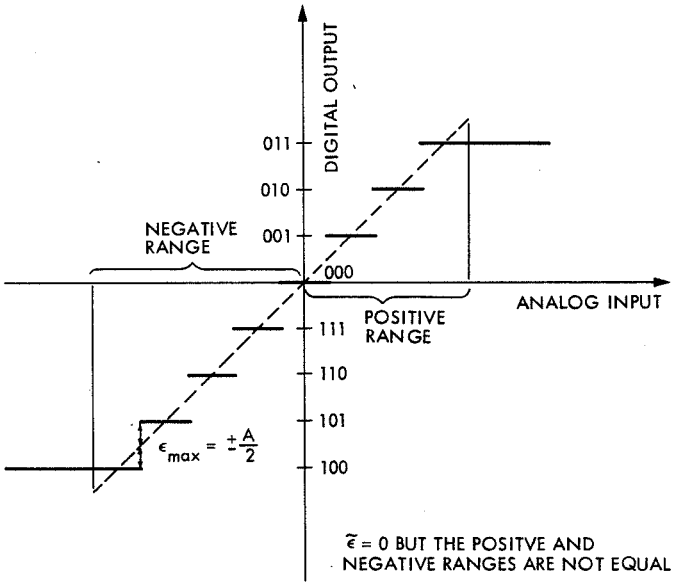


Fig. 6. Output of A-D converter when the offset is zeroed by adjusting its references